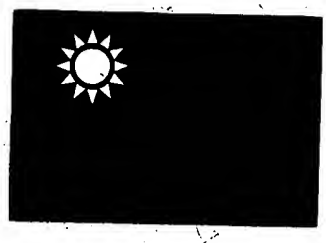
JCLA 6567



Jc973 U.S. PRO
09/920035
08/01/01

中華民國經濟部智慧財產局

INTELLECTUAL PROPERTY OFFICE
MINISTRY OF ECONOMIC AFFAIRS
REPUBLIC OF CHINA

茲證明所附文件，係本局存檔中原申請案的副本，正確無訛，
其申請資料如下：

This is to certify that annexed is a true copy from the records of this office of the application as originally filed which is identified hereunder:

申　請　日：西元 2001 年 03 月 14 日
Application Date

申 請 案 號：090105912
Application No.

申　請　人：財團法人工業技術研究院
Applicant(s)

CERTIFIED COPY OF PRIORITY DOCUMENT

局　　長
Director General

陳明邦

發文日期：西元 2001 年 3 月
Issue Date

發文字號：**09011004726**
Serial No.

申請日期：	案號：
類別：	

（以上各欄由本局填註）

發明專利說明書

一、發明名稱	中文	建構漸進式模型的方法
	英文	
二、發明人	姓名（中文）	1. 楊舒凱 2. 張勤振 3. 段鼎洲 4. 林明芬
	姓名（英文）	1. 2. 3. 4.
	國籍	1. 中華民國 2. 中華民國 3. 中華民國 4. 中華民國
	住、居所	1. 台南市南區德興路164巷26號 2. 新竹縣竹北市嘉興路123號 3. 高雄縣岡山鎮介壽西路321巷2弄7號 4. 新竹市光華二街72巷38之3號8樓之2
三、申請人	姓名（名稱）（中文）	1. 財團法人工業技術研究院
	姓名（名稱）（英文）	1. INDUSTRIAL TECHNOLOGY RESEARCH INSTITUTE
	國籍	1. 中華民國
	住、居所（事務所）	1. 新竹縣竹東鎮中興路四段195號
	代表人姓名（中文）	1. 翁政義
	代表人姓名（英文）	1.

四、中文發明摘要　（發明之名稱：建構漸進式模型的方法）

一種建構漸進式模型的方法，使用森林叢集演算法以及各種誤差評估法，用以維持高度近似原始模型的品質。其能夠在單一步驟中縮減任意數量的頂點或三角形，以製造出能十分平滑且快速地改變精細度的漸進式模型（progressive mesh），適用於網路傳輸(network transmission)或即時描繪(real-time rendering)。

英文發明摘要　（發明之名稱：）

本案已向

國(地區)申請專利	申請日期	案號	主張優先權
無			

有關微生物已寄存於

寄存日期	寄存號碼
無	

五、發明說明（1）

發明的領域

本發明是有關於一種建構漸進式模型的方法，且特別有關於一種使用森林叢集方式而形成的漸進式模型建構方法。

發明的背景

近幾年來的虛擬實境軟體在畫面品質上要求不斷提升，雖然硬體進步突飛猛進，但仍遠遠地追趕不上人們對畫面品質的要求。於是軟體設計者廣泛地使用多精細度描繪技術(Levels of Detail)，也就是說，在場景中，距離觀察者越遠的物體使用越粗糙的模型去描繪，以期在不損傷輸出影像品質的前提下盡量減輕繪圖硬體的負擔。換個角度想，用了這個技術之後，在相同的硬體描繪能力之下，便可以描繪更複雜的場景。如第1A~1C圖中不同精細度的街燈模型，如果在中距離下使用(b)精細度、在遠距離使用(c)精細度，而不是各種距離下都使用(a)精細度來描繪，雖然此種表示方式對於影像品質毫無影響，但是給硬體的負擔卻有相當大的輕重分別。

目前廣泛使用的方案是場景中的物體準備多個不同精細度的模型，在描繪畫面的時候依照距離遠近選擇不同的模型，這個作法非常耗費記憶體。而且產生了另一個問

題：即當不同精細度的模型準備的數量不夠多的時候，物體相對觀察者由遠而近(或由近而遠)移動時，觀察者可以很明顯地感覺到場景中的物體不是變精細了，而是被「偷換」了，這是視覺效果上相當嚴重的突兀，理想效果而言應該是要做到讓使用者感覺不出物體正在改變精細度。

因此在這個需求之下，便有了漸進式模型(Progressive Mesh)的概念，這是一種具有多精細度(Multi-Resolution)的模型格式類型，概念上是將模型簡化時的中間資料和簡化後的簡化模型一併儲存。因此，漸進式模型內的資料必須能使模型在描繪時可以平滑地改變精細度、掌握模型精細度，並及時完成模型的精細度調變。

所謂「精細度」一般指模型上的點或面的數目，數目越少則模型的視覺效果月粗糙。譬如說一個有500個面和一個有1000個面的模型，就是係屬於兩不同的「精細度層次」(Resolution Level)的模型，而這兩不同層次模型在精細度上的差異，就稱為該兩層次之間的「誤差」。

下列我們以現今最常用兩種建構漸進式模型技術的解決的方法來作說明：

1. 頂點叢集法(Vertex Clustering)，如第2圖所示將

模型所佔據的空間均分為許多的小細格(Cell)，並在每個細格中依照視覺重要性(Visual Importance)選出代表點(Representative Vertex)，再將每一個細格內的非代表點都合併到該細格的代表點。在頂點合併之後，對模型中的每一個三角形而言，如果有兩個以上的頂點被合併到同一個點，則將這個三角形剔除。我們可以看出頂點叢集法能夠一次剔除大量的頂點或三角形，但是並沒有在模型特徵處特別保留較多量的幾何資料，所以在模型形狀的保留上效果很差，甚至在對於使用在漸進式模型上，更需要大量記憶改變的方式，才能回復到原來狀態。

2. 邊線折疊法 (Edge Collapsing)，其原理是找出模型上剔除後影響形狀改變最小的邊，例如將第3圖左圖中的兩個端點(v_t, v_s)合併成右圖中的一個點 v_s，使得原來(v_t, v_s) 兩側的三角形消失了，這個合併頂點的動作就稱為邊線折疊(Edge Collapsing)，在這個步驟中模型減少了兩個三角形。由於邊線折疊並不是不可逆的動作，只要將第3圖中的 v_s 重新分裂為 v_t 和 v_s，再將被剔除的兩個三角形 (v_t, v_l, v_s) 以及 (v_r, v_t, v_s) 放回去，這個模型便精細化(Refine)到未施行邊線折疊前的狀態，這個步驟稱為頂點分裂(Vertex Split)。

因此，模型可以經由一系列的邊線折疊，簡化成任意精細度的簡化模型和一連串邊線折疊的過程紀錄，且簡化

後的模型也可以依據所紀錄的邊線折疊過程資料，經由相反順序的頂點分裂復原為精細的模型，這就是 Microsoft Research 在 1996 年提出的漸進式模型(Progressive Mesh) 概念。

如第4圖所示，以一個漸進式模型 M 來說，假設我們將它最粗糙的狀態訂為 M^0，經過頂點分裂(精細化)動作 R_1 之後變成狀態 M^1，而 M^1 可經由 R_1 的反運算邊線折疊(粗糙化) C_1 回到狀態 M^0。依此類推，狀態 M^i 經由 R_{i+1} 到達狀態 M^{i+1}，而 M^{i+1} 經由 C_{i+1} 回到狀態 M^i，於是一個具有 n 個 R 運算的漸進式模型 M 共有 n+1 個不同的狀態，或稱有 n+1 個層次的精細度。

再回到第3圖中我們可以看出，一個邊線折疊運算最多只能縮減兩個三角形，所以用這種方法建構出來的漸進式模型一次最多也只能增減兩個三角形。所以邊線折疊法在對模型簡化效果相當良好，能高度保留原始模型的特徵。然而，一次只增減兩個三角形實在太慢了，如果以第1A圖中 8828 個三角形的模型為例，要改變到第1C圖中 500 個三角形的精細度至少要 (8828-500)/2=4164 次的邊線折疊運算才能達成，因此不適用於需要即時描繪的場合，與快速刪除頂點與三角形的圖形切換。

由於，近年來網際網路的進步和電子商務的推展，漸

進式模型技術有了另一個價值。即透過在網路上傳輸模型的方式，電子商務終於可以販賣平面影像不容易展示的商品，而這些通常是價格較昂貴的商品。使用者由網路上下載商家提供的產品模型，並且觀看商家設計的功能示範或是親自操作，以瞭解產品的操作方式和功能。

然而模型資料是比影像更龐大的資料量，所以除了編碼式的資料壓縮之外，更需要將模型製作成多重精細度的漸進式模型，而讓使用者只要接收到資料量很小的粗糙模型，就可以看到產品的雛形並且進行操作，然後再接著傳送補充資料讓模型變精細。如此可以縮短伺服端對客戶端的使用者的回應時間，如果不是使用者要察看的模型可以在資料傳送結束前就中斷傳輸，以節省網路頻寬。因此當使用者在瀏覽多個模型的時候，這個技術更顯得相當重要。

發明概述

上述兩種已知模型簡化方式在使用於漸進式模型上，都有其缺點存在，因此本發明提出一種使用森林叢集之漸進式模型建構方法，能夠利用品質控制方法以及誤差估計法則，以產生高度近似原始模型的簡化模型或漸進式模型，並且能夠滿足網路傳輸與即時描繪的需求。

本發明提供一種使用森林叢集之漸進式模型建構方法，包括下列步驟：首先在步驟(a)時，對一由複數個頂點所構成之單一精細度模型中每一頂點分別建立叢集，並建立每一該頂點合併其鄰近的複數頂點之延伸，且計算該延伸運算之成本。接著，在步驟(b)重複以成本最低的延伸組成森林，實施成本最低的有效延伸運算(i, j, k)，其中i、j、k為模型中之頂點，並以頂點k為合併後叢集的樹根，直到達到第一終止條件為止。然後在步驟(c)時，對上述森林中的每一叢集c(t)，進行一叢集簡化動作，合併非樹根的頂點，其中t為該叢集c(t)之代表點。

最後，我們重複步驟(b)、(c)，直到達到第二終止條件為止，以產生一個使用者需要的簡化模型。

若我們在步驟(c)時，記錄每一次簡化動作為簡化紀錄，則可將簡化紀錄轉化成為回復序列，以供將來需將模型精細化時使用。

為讓本發明之上述和其他目的、特徵、和優點能更明顯易懂，下文特舉較佳實施例，並配合所附圖式詳細說明。

發明的詳細敘述

由於頂點叢集法無法偵測模型表面的變化，所以沒能保留模型特徵，就把空間上相接近的頂點合併。至於邊線折疊法雖確實估量合併兩個頂點的成本而能保留模型特徵，但卻失去頂點叢集法能夠一次剔除大量的幾何資料的效率優勢。

因此本發明所使用森林叢集之漸進式模型建構方法，在去除上述兩項習知技術的缺點下，能夠確實估量合併頂點的成本，且找出的合併範圍是分佈於模型上的邊線森林，一次剔除之，所以能夠兼具頂點叢集法和邊線折疊法的優點。例如第5圖所示，將在左圖由 5030 個三角形所組成的貝多芬雕像找出呈森林分佈的可合併頂點，一次剔除 2012 個三角形，成為右圖的簡化後模型。以下我們將詳細說明本發明一實施例。

（一）延伸的成本評估(Cost Evaluation of an Expansion)：對輸入之單一精細度層次的模型中每一個點建立叢集，對每一個點及其鄰近的頂點建立延伸，並進行合併成本評估，且以此推導出將多個頂點合併到一個頂點的延伸運算成本。

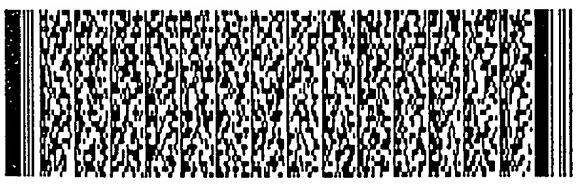
（二）生成森林(Forest Growth)：重複以成本最低的延伸運算(u, v, v_0) 生成森林，直到達到該回合的終止條件為止。並在延伸運算的過程中，以合併者v_0為代表點

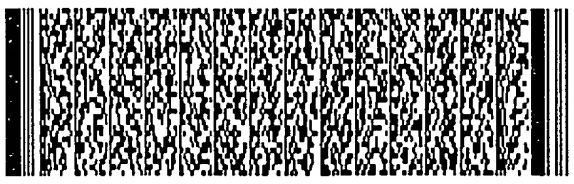
(Representative Vertex)，亦即該棵頂點樹的樹根。

（三）進行叢集(Performing Clustering)簡化動作：對於生成的森林中的每棵樹，將樹中的非樹根點都合併到樹根頂點。然後，繼續進行進一步簡化動作時，對於有一個頂點被合併的三角形，將三角形被合併的那頂點替換為該叢集的代表點。去除模型中有兩個頂點以上被合併到同一個頂點的三角形。同時，可以將作過的簡化動作一一記憶起來。

如此重複（二）、（三）的步驟，直到滿足使用者指定之終止條件為止，便產生使用者所需要的簡化模型。並且可將所儲存的簡化動作轉換為一序列對應的回復動作(Refinement Sequence)。

首先，輸入之單一精細度層次的模型係由複數個點(頂點)形成複數個三角形所構成，我們以一個頂點叢集(Vertex Cluster)作為模型頂點的子集合，當中包括了一個代表點(Representative Vertex)，叢集中其他的頂點都預定會被合併到這個代表點，以完成模型簡化。本發明方法中的頂點叢集不但是模型頂點的子集合，而且是模型拓樸上一棵頂點樹(Vertex Tree)中的節點，叢集中的代表點就是此叢集之樹根(Root)。





我們以c(x)表示以頂點x為代表點的頂點叢集(Vertex Cluster)，以圖6為例，c(u)和c(v_0)分別是以u點和v_0點為代表點的頂點叢集，v是叢集c(v_0)中的一個頂點，而延伸(Expansion) e=(u, v, v_0)是透過連接邊線(u, v)將叢集c(u)合併到叢集c(v_0)的運算，延伸e的成本是對叢集c(u)中所有的頂點計算合併到頂點v_0的成本後取最大值，而所謂實施延伸e(Apply Expansion e)也就是將線(u, v)連接起來。所以我們接著對模型中每一點所鄰近的頂點，開始推算出每一點合併其每一鄰近的點之成本，用以進行延伸成本評估。

而推算合併兩個頂點的成本(或稱為誤差成本)時，有不同的定義方式。例如：有以合併前後頂點的移動距離為成本，因此選擇模型上最短的邊來折疊，直到模型上的邊都大於給定的容忍值；有以合併前後頂點偏離所在三角形的平面的垂直距離平方和以合併前後頂點鄰近三角形平面法向量的轉動角度為成本，因此選擇模型上兩端點最接近共平面的邊來折疊。

也有些作法乾脆不給定容忍值，重覆邊線折疊的動作直到三角形數目下降到使用者給定的目標。另外，有人提出以矩陣形式來定義成本的方式，定義一個頂點v的誤差為「折疊前後頂點v偏離所在三角形的平面的垂直距離平方和」，而合併一條邊線時兩端點造成的誤差和就是

折疊這條邊線的成本。

同時，使用者依據其所需要的漸進式模型簡化目標，指定下述條件，以建構出符合使用者指定的漸進式模型：

1. 層次與層次之間的步進條件，以推導出每回合森林生成的終止條件。例如在精細化時，每精細度層次增加的三角形數量為上一層次的10%，或是誤差縮減為上一層次的90%。而系統則依據此步進條件，推導出「每一簡化回合的終止條件」（第一終止條件），例如在本回合中預定要減少的三角形數量或是本回合的誤差上限。

2. 所需最粗糙模型(Base Mesh)條件，以作為「模型簡化的終止條件」（第二終止條件）。例如：三角形數量少於某個指定的數量，或是簡化模型與原始模型的誤差到達某個上限，或是精細度層次數量到達某個數量。

接著，我們要進行生成森林的步驟。每實施一個延伸(Expansion)運算會連接一條模型上的一條邊(Edge)，而在每個回合裡，系統不斷地進行延伸運算，直到達到本回合的終止條件（第一終止條件），連接這些邊會形成一個森林(Forest)，對森林中的每棵樹來說，樹根頂點(Root Vertex)稱為樹中所有頂點的代表點(Representative Vertex)。

本發明本著「適者生存」的精神，並將被合併所需成本較高之頂點，留下來當代表點。首先，準備一個延伸運算的儲列，在儲存時依照成本將延伸由小到大存放，初始的狀態下是空的。然後為模型的每一個頂點 x 都建立一個叢集 c(x)，每個叢集 c(x) 中除了 x 之外再沒有其他的頂點。對頂點 x 每一個鄰近點 y 建立延伸 (y, x, x)，推算其成本之後放到一個儲列中。當然，在儲列中也存在延伸 (x, y, y)，而在生成森林的時候，如果實施延伸 (y, x, x) 運算的成本比延伸 (x, y, y) 高，那麼結果就是建立延伸 (x, y, y) 且將延伸 (y, x, x) 作廢。

在運作時，重複地自儲列中取出成本最低之延伸，如果這個延伸是尚未被作廢的有效延伸，則實施這個延伸運算，以將此延伸建立起來。如此，不斷以成本最低的延伸運算組成森林。以第6圖為例，如果取出的延伸是 (u, v, v_0)，則將叢集 c(u) 連接在叢集 c(v_0) 中的 v 點，使叢集 c(u) 成為叢集 c(v_0) 的子樹(sub-tree)，此時叢集 c(u) 已經被叢集 c(v_0) 合併、已經不存在了，所以儲列中所有要合併叢集 c(u) 的延伸(u, *, *)全都必須作廢，且將叢集 c(u) 去合併其他叢集的延伸 (*, *, u) 改寫為由叢集 c(v_0) 去合併的延伸 (*, *, v_0)，其中符號 * 代表任意頂點。

延伸（u, v, v_0）被作廢是指下面情形時：若頂點u 已經被合併到某個叢集c(x) 或者延伸（v, u, x）已經被實施過時，則將延伸（u, v, v_0）丟棄(成為無效)；若叢集c(v_0) 在建立延伸（u, v, v_0）之後還曾經合併其他的頂點，則重新計算延伸（u, v, v_0）的成本後，放回到儲列中；若頂點v_0 已經被合併到叢集c(w)，則將延伸（u, v, v_0）改變為（u, v, w），並重新計算延伸成本後，將延伸放回到儲列中，其中x、w 代表任意異於v_0的頂點。

如此藉由不斷的合併叢集來生成森林，在合併的同時統計合併後會被剔除的三角形數量，或者是統計合併後會造成的誤差，以此決定是否停止這一回合的森林長成，每個簡化回合會產生一層精細度，所以本發明產生的漸進式模型可以掌握每一層精細度的幾何資料量或誤差。

然後，我們進行叢集動作(Perform Clustering)，對於生成的森林中的每棵樹，將樹中的非樹根點都合併到樹根頂點，於是減少的頂點數量就是各樹中的非樹根頂點數量總和。除了自己之外還包含其他頂點的頂點叢集(Vertex Cluster)就是要進行叢集簡化動作的對象，其動作即將樹根合併其延伸的非樹根之頂點。然後，若三角形上只有一個頂點被合併時，將被合併之頂點以該頂點之代表點取代，不刪除此三角形。如果一個三角形有兩個以上的頂點被合併，但不是被合併到同一個頂點時，也視同前

述狀況，將那些被合併的頂點替換為各自的代表點。而當三角形上至少有兩個頂點被合併到同一個頂點時，由於此三角形已被摺疊，則將此三角形刪除。

如第7圖就是處理第6圖產生的叢集$c(v_0)$，如前所述，每一個叢集是一株頂點樹(Vertex Tree)，對叢集$c(v_0)$進行叢集動作，係進行下列三件事：

1. 將樹根所延伸的樹幹上的該些點合併到樹根。在運作上，即將圖形內樹中頂點 v_0 以外的每一個頂點都合併到頂點 v_0。

2. 去除有樹幹經過三角形兩個頂點以上的三角形。在運作上，即將所有使用到$c(v_0)$中的頂點達到兩個以上的三角形剔除，因為他們有至少兩個的點被叢集了。

3. 將有樹幹經過三角形之一個頂點的角落移動到樹根上，以形成一簡化模型。在運作上，對只有使用一個$c(v_0)$中的頂點且非頂點 v_0 的三角形，將它使用該頂點的角落移動到頂點 v_0 的位置，並不剔除之，以封閉模型上刪除三角形所產生的缺口。

如此，對森林中的每一株頂點樹都做這樣的處理之後，所以每一個叢集又回到「只包含一個代表點不包含其他頂點」的狀態，因此又可以進行下一回合的模型簡化。同時，可以將上述進行叢集所進行之簡化動作記憶下來，

成為簡化紀錄。

一個簡化回合包含森林生成與進行叢集兩個步驟，每回合後檢查是否已經達到使用者指定的模型簡化終止條件，若尚未達到則繼續進行下一簡化回合。即以此簡化模型重複進行上述動作，直到一使用者需求之最粗糙模型形成。同時，將上述記憶下的簡化紀錄，轉換成為一序列的回復動作(Refinement Sequence)紀錄與簡化後的模型一起儲存。

而且，我們可以最粗糙模型為基礎，並根據簡化紀錄中所記憶之刪除一些頂點、刪除一些三角形、移動一些三角形的某些角落的動作，轉換成對應的回復序列，以反向動作復到上一個簡化模型。要將簡化後的模型精細化一個層次只要以簡化時期的相反順序：將這回合裡被移動的三角形角落移回原位；加入原本在這回合裡被刪除的頂點和三角形。如此我們只要將一序列的回復動作紀錄和簡化後的模型一併儲存，就能夠以漸進方式高效率地變換模型精細度。

在應用上，如對 MPEG-4 標準的支援 (Supporting the 3D Mesh Object Specification of MPEG-4)的相關文件中指出，只要是這種叢集法類型的，都會在每個回合後產生一個「刪除的三角形集合」的演算法，而可以產生

MPEG-4 規格的漸進式模型。本發明所提出的方法只要在生成森林的時候實行 MPEG-4 文件描述的拓樸測試，以避免在合併頂點的時候產生所謂複雜的曲面(Non-manifold Curve)，避免在刪除的三角形區域內產生內部頂點(Internal Vertex)，那麼每個回合所剔除的三角形就可以拼湊成數段三角形條，並且套用漸進式分裂壓縮技術儲存為 MPEG-4 標準的檔案格式。

因此本發明所提出的漸進式模型具有以下特色與優點：

1. 保留原始模型的幾何與屬性特徵。

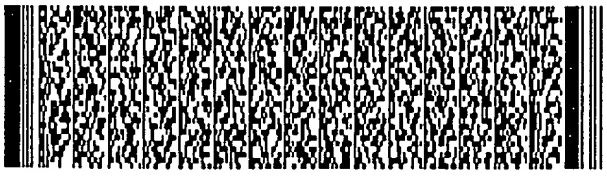
2. 能夠掌握每一層精細度的資料量或誤差，在調變精細度時保證視覺效果的平滑。

3. 能夠有效率地調變精細度，在描繪時即時完成模型的精細化或粗糙化。

所以，在分散式虛擬環境(Distributed Virtual Environment)中，模型可能透過網路傳輸到客戶端之後，馬上就要做多精細度即時描繪用，使得本發明所提出的能夠兼顧網路傳輸和即時描繪需求的的漸進式模型。特別是電子商務中需要傳輸模型以展示產品功能，需要具備讓使用者短時間內瀏覽多個模型的功能，或者遊戲軟體中的即時描繪，以利於在有限的硬體能力之下描繪更廣大、複雜的場景，使遊戲能夠呈現更精緻、流暢的視覺效果。另

外，在模擬平台下的即時描繪，例如 CAVE 系統，需要在運作時即時描繪多個大型的畫面，故硬體負擔相當繁重，運用本發明可以省下不必要的硬體負擔，進而能夠描繪更複雜的場景，呈現更精緻的畫面。

雖然本發明已以較佳實施例揭露如上，然其並非用以限定本發明，任何熟習此技藝者，在不脫離本發明之精神和範圍內，當可作各種之更動與潤飾，因此本發明之保護範圍當視後附之申請專利範圍所界定者為準。



圖式簡單說明

第1A~1C圖繪示不同精細度的街燈模型；

第2圖繪示頂點叢集法的運作；

第3圖繪示邊線折疊法的運作；

第4圖是一般化漸進式模型示意圖；

第5圖繪示使用本發明方法將5030個三角形所組成的貝多芬雕像，一次剔除2012個三角形所形成的簡化模型；

第6圖繪示頂點叢集$c(u)$、$c(v_0)$與延伸$e=(u, v, v_0)$。

第7圖繪示對第6圖進行森林叢集的運作。

六、申請專利範圍

1. 一種建構漸進式模型的方法，包括下列步驟：

（a）對一由複數個頂點所構成之單一精細度模型中每一頂點分別建立叢集，並建立每一該頂點合併其鄰近的複數頂點之延伸，且計算該延伸運算之成本；

（b）重複以成本最低的延伸組成森林，實施成本最低的有效延伸運算(i, j, k)，其中 i、j、k 為模型中之頂點，並以頂點 k 為合併後叢集的樹根，直到達到第一終止條件為止；

（c）對上述森林中的每一叢集 c(t)，進行一叢集簡化動作，合併非樹根的頂點，其中 t 為該叢集 c(t) 之代表點；且

（d）重複步驟(b)、(c)，直到達到第二終止條件為止，以產生一簡化模型。

2. 如申請專利範圍第1項所述之建構漸進式模型的方法，其中組成森林之步驟，復包括下列步驟：

取得一成本最低的延伸 (u, v, v_0)，其中 u、v、v_0 為該單一精細度模型中之頂點；及

若該頂點 u 已經被合併到一叢集 c(x) 或者一延伸 (v, u, x) 已經被實施過時，則設定該延伸 (u, v, v_0) 無效，其中 x 為該單一精細度模型中異於 v_0 的任意頂點。

3. 如申請專利範圍第2項所述之建構漸進式模型的方法

，其中組成森林之步驟，復包括一步驟：

若叢集 $c(v_0)$ 在建立該延伸(u, v, v_0) 之後還曾經合併其他的頂點，則重新計算該延伸(u, v, v_0) 的成本，並不實施該延伸。

4. 如申請專利範圍第2項所述之建構漸進式模型的方法，其中組成森林之步驟，復包括一步驟：

若該頂點 v_0 已經被合併到一叢集 $c(w)$，則將該延伸(u, v, v_0) 改變為 (u, v, w)，並重新計算成本，並不實施該延伸(u, v, v_0)，其中 w 為該單一精細度模型中異於 v_0 的任意頂點。

5. 如申請專利範圍第3項所述之建構漸進式模型的方法，其中組成森林之步驟，復包括一步驟：

若該頂點 v_0 已經被合併到一叢集 $c(w)$，則將該延伸(u, v, v_0) 改變為 (u, v, w)，並重新計算成本，並不實施該延伸(u, v, v_0)，其中 w 為該單一精細度模型中異於 v_0 的任意頂點。

6. 如申請專利範圍第2項所述之建構漸進式模型的方法，其中在進行叢集簡化動作時，將該代表點 t 以外的每一個頂點都合併到該代表點 t。

7. 如申請專利範圍第6項所述之建構漸進式模型的方法

，其中構成該精細度模型之複數個點形成複數個三角形，且其中進行叢集簡化動作之步驟復包括一步驟：將所有使用到該叢集c(t)中的頂點達到兩個以上的三角形剔除。

8. 如申請專利範圍第6項所述之建構漸進式模型的方法，其中構成該精細度模型之複數個點形成複數個三角形，且其中進行叢集簡化動作之步驟復包括一步驟：對只有使用該叢集c(t)中的一個頂點且非代表點t的三角形，將它使用該頂點的角落移動到該代表點t的位置。

9. 如申請專利範圍第7項所述之建構漸進式模型的方法，其中進行叢集簡化動作之步驟復包括一步驟：對只有使用叢集c(t)中的一個頂點且非代表點t的三角形，將它使用該頂點的角落移動到該代表點t的位置。

10. 如申請專利範圍第5項所述之建構漸進式模型的方法，其中在進行叢集簡化動作時，將該代表點t以外的每一個頂點都合併到該代表點t。

11. 如申請專利範圍第10項所述之建構漸進式模型的方法，其中構成該精細度模型之複數個點形成複數個

三角形，且其中進行叢集簡化動作之步驟復包括下列步驟：

將所有使用到叢集c(t)中的頂點達到兩個以上的三角形剔除；以及

對只有使用叢集c(t)中的一個頂點且非代表點t的三角形，將它使用該頂點的角落移動到該代表點t的位置。

12. 如申請專利範圍第1項所述之建構漸進式模型的方法，其中進行叢集簡化動作之步驟，復包括一步驟：

將每一次所作之簡化動作，作成一簡化記錄。

13. 如申請專利範圍第12項所述之建構漸進式模型的方法，復包括一步驟：

將該簡化紀錄轉化成對應的一回復動作紀錄。

14. 如申請專利範圍第9項所述之建構漸進式模型的方法，其中進行叢集簡化動作之步驟，復包括一步驟：

將每一次所作之簡化動作，作成一簡化記錄。

15. 如申請專利範圍第14項所述之建構漸進式模型的方法，復包括一步驟：

將該簡化紀錄轉化成對應的一回復動作紀錄。

六、申請專利範圍

16. 如申請專利範圍第11項所述之建構漸進式模型的方法，其中進行叢集簡化動作之步驟，復包括一步驟：
將每一次所作之簡化動作，作成一簡化記錄。

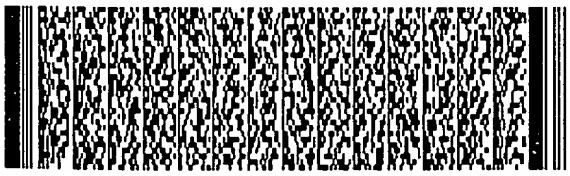
17. 如申請專利範圍第16項所述之建構漸進式模型的方法，復包括一步驟：
將該簡化紀錄轉化成對應的一回復動作紀錄。

18. 如申請專利範圍第1項所述之建構漸進式模型的方法，其中之第一終止條件推導自使用者指定的精細度模型層次與層次間的一步進條件。

19. 如申請專利範圍第1項所述之建構漸進式模型的方法，其中之第二終止條件係一使用者指定的最粗糙模型條件。

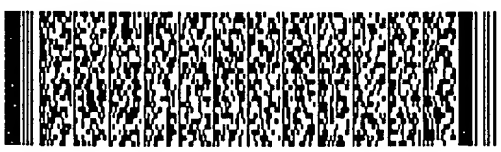
20. 如申請專利範圍第19項所述之建構漸進式模型的方法，其中之第二終止條件係一使用者指定的最粗糙模型條件。

21. 如申請專利範圍第13項所述之建構漸進式模型的方法，其中之第一終止條件推導自使用者指定的精細度模型層次與層次間的一步進條件，且第二終止條件係一使用者指定的最粗糙模型條件。



22. 如申請專利範圍第15項所述之建構漸進式模型的方法，其中之第一終止條件推導自使用者指定的精細度模型層次與層次間的一步進條件，且第二終止條件係一使用者指定的最粗糙模型條件。

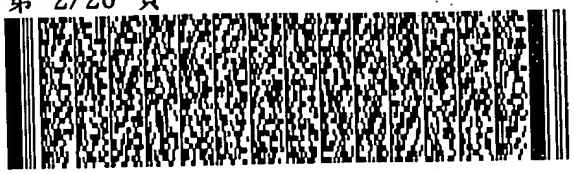
23. 如申請專利範圍第17項所述之建構漸進式模型的方法，其中之第一終止條件推導自使用者指定的精細度模型層次與層次間的一步進條件，且第二終止條件係一使用者指定的最粗糙模型條件。



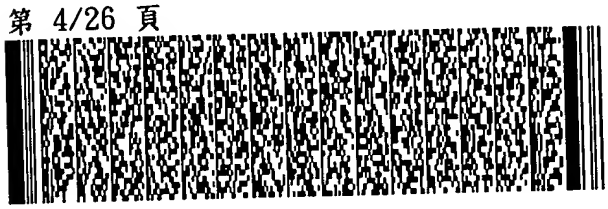
申請案件名稱 :建構漸進式模型的方法

第 1/26 頁

第 2/26 頁



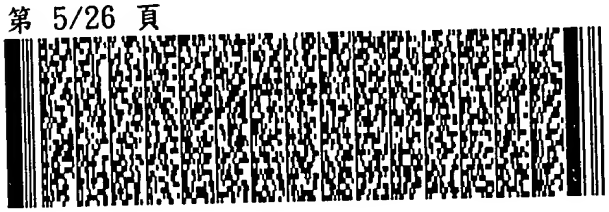
第 4/26 頁



第 4/26 頁

第 5/26 頁

第 5/26 頁



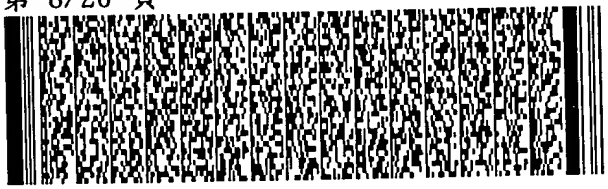
第 6/26 頁

第 6/26 頁

第 7/26 頁

第 7/26 頁

第 8/26 頁

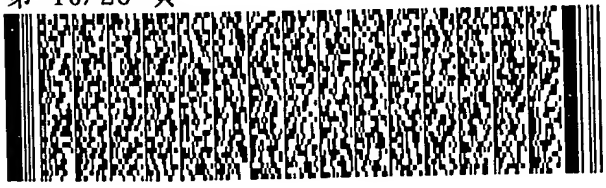


第 8/26 頁

第 9/26 頁

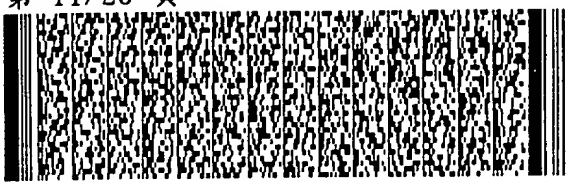
第 9/26 頁

第 10/26 頁

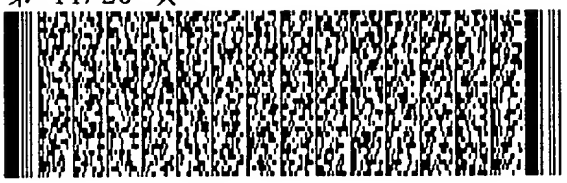


第 10/26 頁

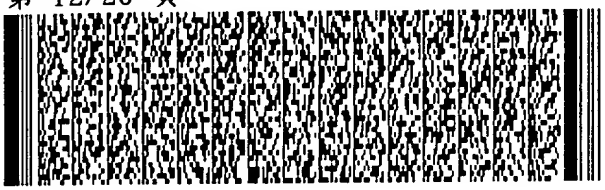
第 11/26 頁



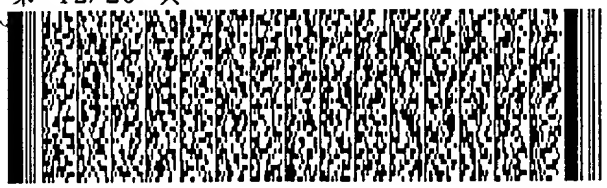
第 11/26 頁



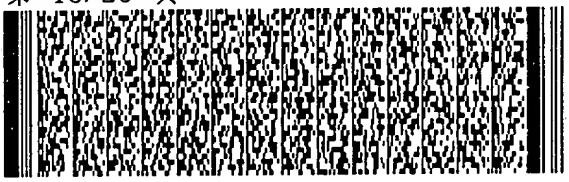
第 12/26 頁



第 12/26 頁

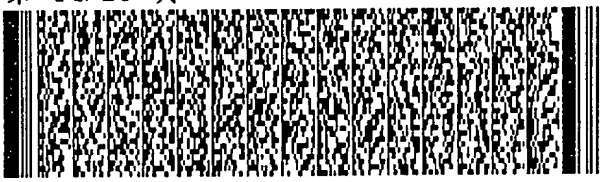


第 13/26 頁

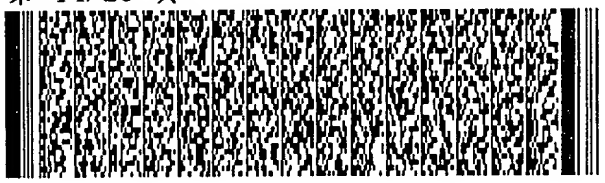


第 13/26 頁

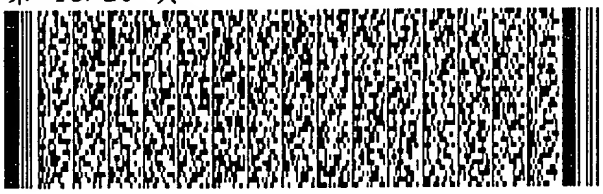
第 14/26 頁



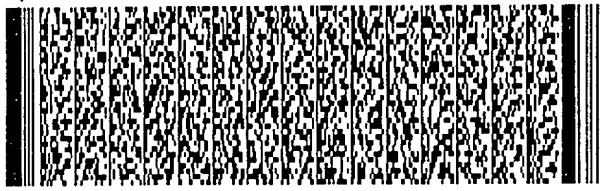
第 14/26 頁



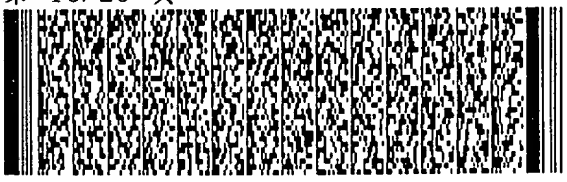
第 15/26 頁



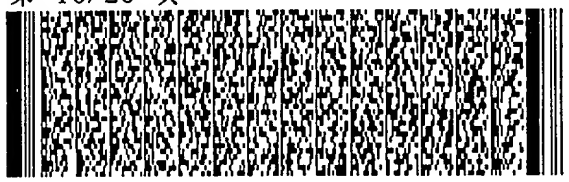
第 15/26 頁



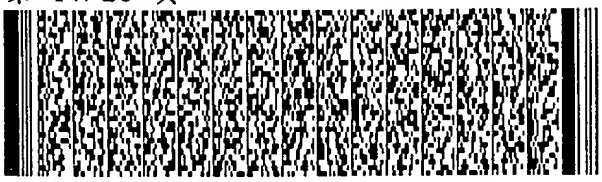
第 16/26 頁



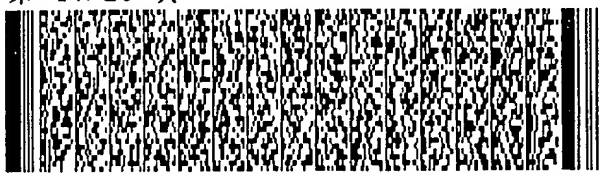
第 16/26 頁



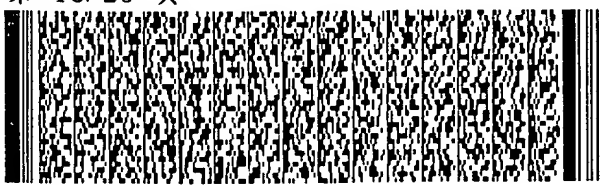
第 17/26 頁



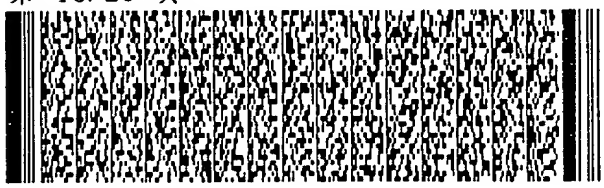
第 17/26 頁



第 18/26 頁

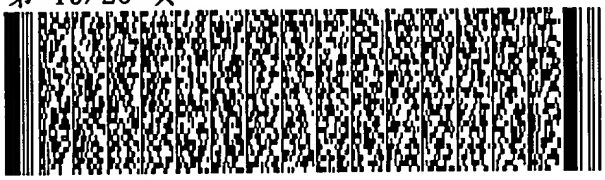


第 18/26 頁



申請案件名稱:建構漸進式模型的方法

第 19/26 頁

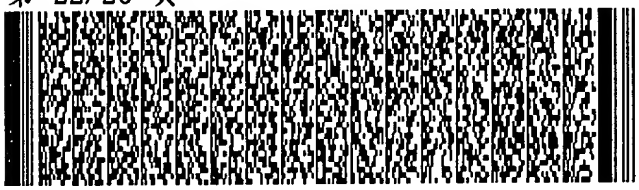


第 20/26 頁

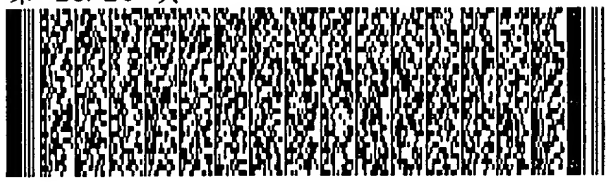
第 21/26 頁

第 21/26 頁

第 22/26 頁

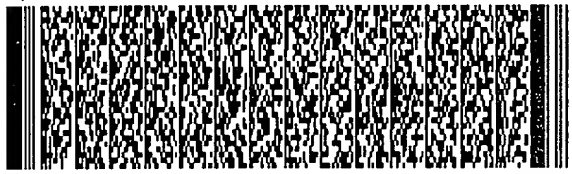


第 23/26 頁

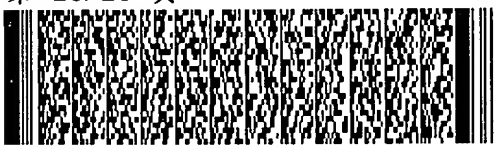


第 24/26 頁

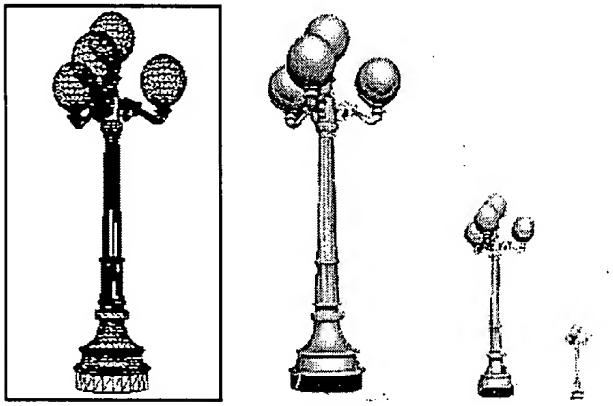
第 25/26 頁



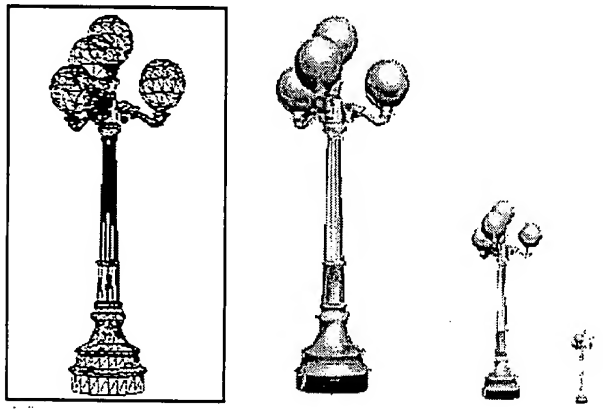
第 26/26 頁



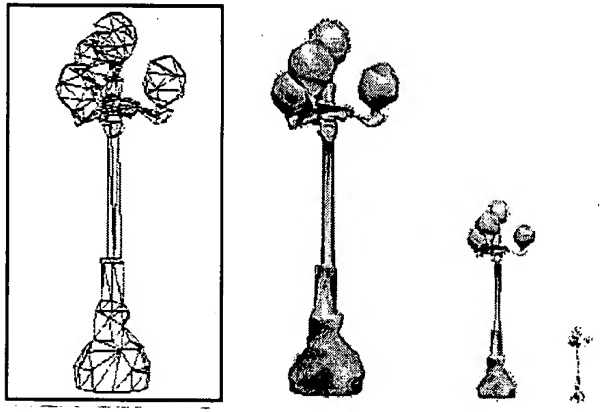
6567TW



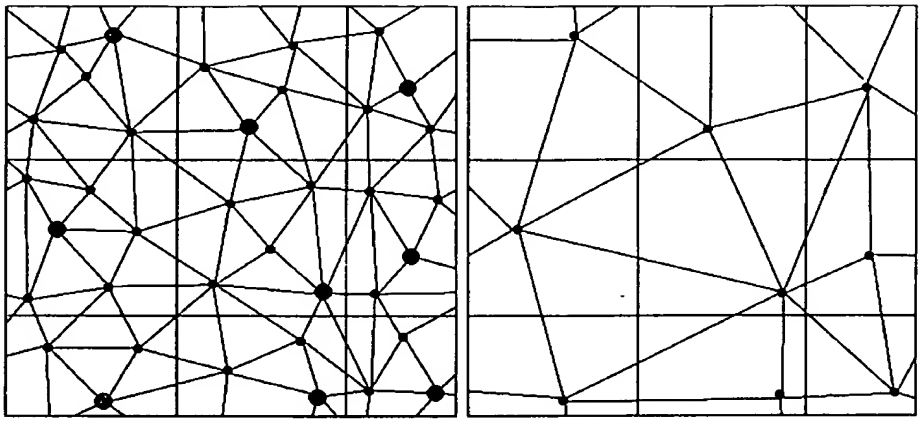
第1A圖



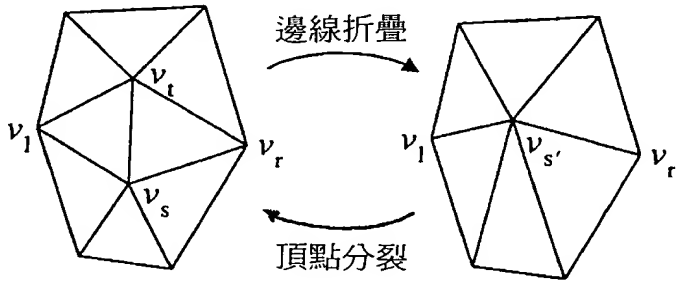
第1B圖



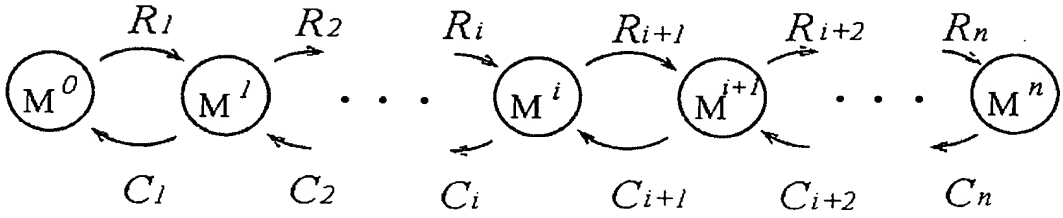
第1C圖



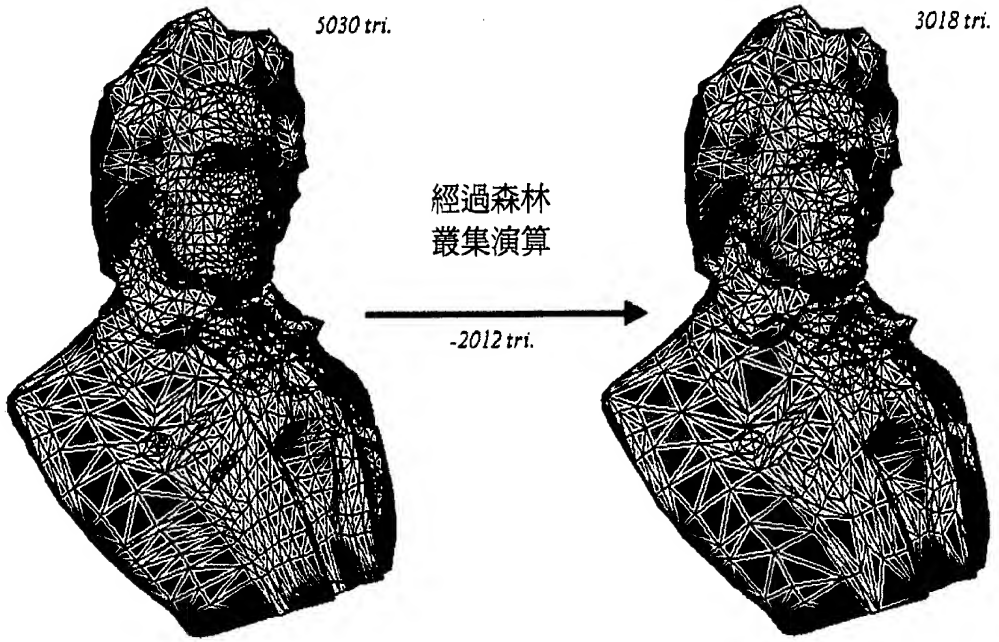
第 2 圖


邊線折疊
v_t
v_l
v_r
v_s
v_l
$v_{s'}$
v_r
頂點分裂

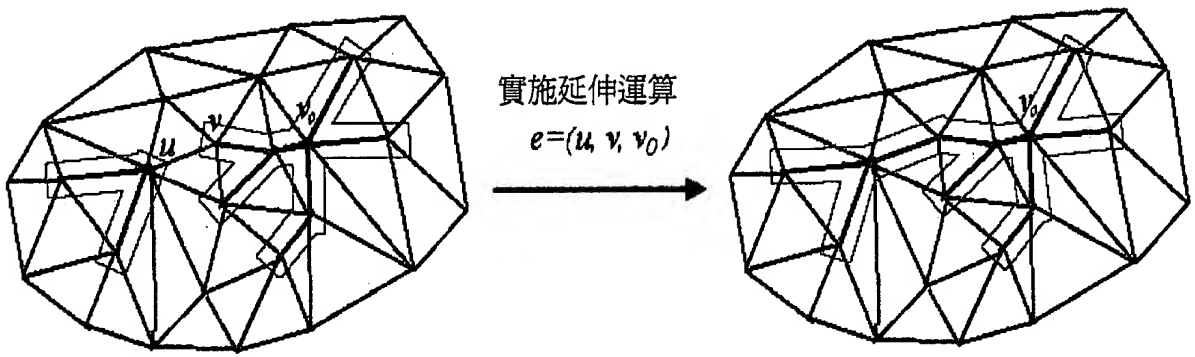
第 3 圖


R_1
R_2
R_i
R_{i+1}
R_{i+2}
R_n
M^0
M^1
M^i
M^{i+1}
M^n
C_1
C_2
C_i
C_{i+1}
C_{i+2}
C_n

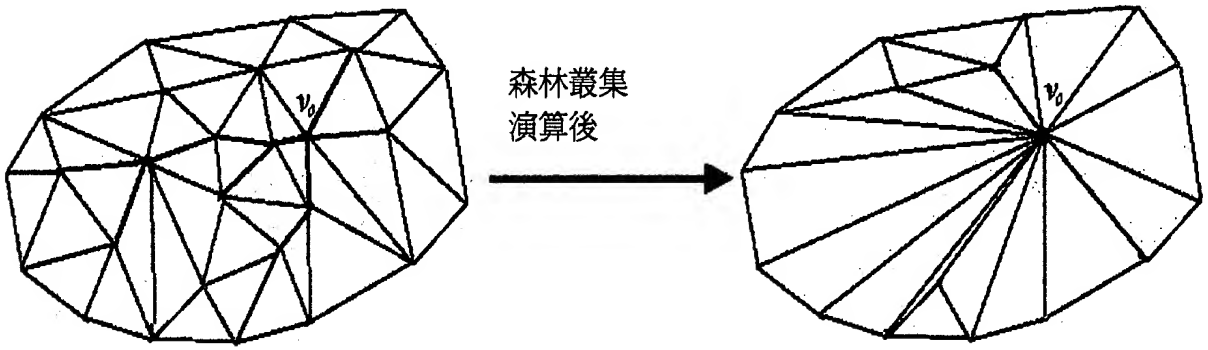
第 4 圖



第 5 圖



第 6 圖



第 7 圖